Appendix A

CaptureDriver.cpp

```
#include "stdafx.h"

#include "CaptureDriver.h"
#include "registryutil.h"
#include "FileUtil.h"
#include "GlobalVariables.h"


BOOL CheckCaptureStart( void )
{
        CString strWaveFolder                    =
::RegGetWaveDirectory();
        ::AppendBackSlashToPath( strWaveFolder );
        CString strFirstCapturedWave     = strWaveFolder + _T(
"snd00000.dat" );
        FILE    *fp;
        fp                 = ::fopen( strFirstCapturedWave.GetBuffer( 0
), "r" );
        if ( fp == NULL )
        {
                return FALSE;
        }
        ::fclose( fp );
        return TRUE;
}

void StartCaptureDriver( void )
{
        /*
        FILE    *fp;
        fp = ::fopen( "C:\\iDroid\\sndcap\\sndcap.ord", "wb" );
        ::fclose( fp );
        */
        ::MakeNewFolder( g_strWaveFolder );
        ::RegWriteCaptureStatus( CString( "on" ) );
}

void StopCaptureDriver( void )
{
        //::DeleteFile( "C:\\iDroid\\sndcap\\sndcap.ord" );
        ::RegWriteCaptureStatus( CString( "off" ) );

        // Beep
        //::MessageBeep( MB_OK );

        ConvertToWave( 32000 );
}
```

```
BOOL MoveCapturedWave( CString strDestinationPath )
{
        ::DeleteFile( strDestinationPath.GetBuffer( 0 ) );
        //return ( ::MoveFile( "C:\\iDroid\\sndcap\\sndcap.wav",
strDestinationPath.GetBuffer( 0 ) ) );
        CString strCapturedWave = ::RegGetWaveDirectory();
        ::AppendBackSlashToPath( strCapturedWave );
        strCapturedWave                 += _T( "sndcap.wav" );
        return ::MoveFile( strCapturedWave.GetBuffer( 0 ),
strDestinationPath.GetBuffer( 0 ) );
}

void ConvertToWave( UINT iFreq )
{
        FILE *pFile;
        char buff[65535];
        char convbuff[65535];
        int     buffsize;
        ULONG bufflen;
        ULONG convbuffsize;
        int fileNum;
        char fileName[ 256 ];

        WAVEFORMATEX WaveFormatEx;
        HMMIO hFile;
        //MMCKINFO MMCKInfoData;
        MMCKINFO MMCKInfoParent;
        MMCKINFO MMCKInfoChild;

        PWRITESOUNDFILE pWriteSoundFile= (PWRITESOUNDFILE) new
WRITESOUNDFILE;

        memset(&WaveFormatEx,0x00,sizeof(WaveFormatEx));
        WaveFormatEx.wFormatTag = WAVE_FORMAT_PCM;
        WaveFormatEx.nChannels = 2;
        WaveFormatEx.wBitsPerSample = 16;
        WaveFormatEx.cbSize = 0;
        WaveFormatEx.nSamplesPerSec = iFreq;
        WaveFormatEx.nAvgBytesPerSec =
WaveFormatEx.nSamplesPerSec*(WaveFormatEx.wBitsPerSample/8)*WaveFormat
Ex.nChannels;
        WaveFormatEx.nBlockAlign =
(WaveFormatEx.wBitsPerSample/8)*WaveFormatEx.nChannels;

        ZeroMemory(pWriteSoundFile,sizeof(WRITESOUNDFILE));
        char *p = pWriteSoundFile->lpszFileName;
        //strcpy(p,"c:\\iDroid\\sndcap\\sndcap.wav");
        // by tuttii
```

```
	CString strWaveFolder   = ::RegGetWaveDirectory();
	::AppendBackSlashToPath( strWaveFolder );
	CString strCapturedWave = strWaveFolder + _T( "sndcap.wav" );
	strcpy( p, strCapturedWave.GetBuffer( 0 ) );


memcpy(&pWriteSoundFile->waveFormatEx,&WaveFormatEx,sizeof(WaveFormatE
x));

	int cbWaveFormatEx = sizeof(WAVEFORMATEX) +
pWriteSoundFile->waveFormatEx.cbSize;

	hFile = ::mmioOpen(pWriteSoundFile->lpszFileName,NULL,
MMIO_CREATE|MMIO_WRITE|MMIO_EXCLUSIVE | MMIO_ALLOCBUF);
	if(!hFile) return;

	ZeroMemory(&MMCKInfoParent, sizeof(MMCKINFO));
	MMCKInfoParent.fccType = mmioFOURCC('W','A','V','E');

	MMRESULT mmResult =  ::mmioCreateChunk( hFile,&MMCKInfoParent,

MMIO_CREATERIFF);

	ZeroMemory(&MMCKInfoChild, sizeof(MMCKINFO));
	MMCKInfoChild.ckid = mmioFOURCC('f','m','t',' ');
	MMCKInfoChild.cksize = cbWaveFormatEx;
	mmResult = ::mmioCreateChunk(hFile, &MMCKInfoChild, 0);
	mmResult = ::mmioWrite(hFile,
(char*)&pWriteSoundFile->waveFormatEx, cbWaveFormatEx);
	mmResult = ::mmioAscend(hFile, &MMCKInfoChild, 0);
	MMCKInfoChild.ckid = mmioFOURCC('d', 'a', 't', 'a');
	mmResult = ::mmioCreateChunk(hFile, &MMCKInfoChild, 0);

	fileNum = 0;
	//sprintf(fileName, "C:\\iDroid\\sndcap\\snd%05d.dat",
fileNum);
	// by tuttii
	sprintf(fileName, "%ssnd%05d.dat", strWaveFolder.GetBuffer( 0
), fileNum);

	while( (pFile = fopen(fileName, "rb")) != 0 )
	{
		while ( fread( &bufflen, 1, 4, pFile ) != 0 )
		{
			if ( bufflen > 65535 ) OutputDebugString( "out
of buffer" );

			if( ( buffsize = fread( buff, 1, bufflen,
pFile) ) != 0 )
```

```
                                {
                                        //convbuffsize = PCMconvert( buff,
convbuff, buffsize );
                                        //convbuffsize = ConvertRate( buff,
convbuff, buffsize );
                                        convbuffsize = PCMconvert2( buff,
convbuff, buffsize );
                                        //convbuffsize = PCMNoConvert( buff,
convbuff, buffsize );
                                        ::mmioWrite(hFile, convbuff,
convbuffsize);
                                }
                        }
                        fclose( pFile );
                        DeleteFile( fileName );
                        fileNum++;
                        sprintf(fileName, "%ssnd%05d.dat",
strWaveFolder.GetBuffer( 0 ), fileNum);
                }

                if(hFile)
                {
                        ::mmioAscend(hFile, &MMCKInfoChild, 0);
                        ::mmioAscend(hFile, &MMCKInfoParent, 0);
                        ::mmioClose(hFile, 0);
                        hFile = NULL;
                }

                return;
}

ULONG PCMconvert( char* buff, char* convbuff, ULONG buffsize )
{
        ULONG buffptr = 0;
        ULONG convbuffptr = 0;

        if ( ( buffsize > 440-20 ) && ( buffsize < 440+20 ) ) // 11k
        {
                while( buffptr+1 < buffsize )
                {
                        *(convbuff + convbuffptr + 0 )= *(buff +
buffptr + 0 );
                        *(convbuff + convbuffptr + 1 )= *(buff +
buffptr + 1 );
                        *(convbuff + convbuffptr + 2 )= *(buff +
buffptr + 2 );
                        *(convbuff + convbuffptr + 3 )= *(buff +
buffptr + 3 );
```

```
                        *(convbuff + convbuffptr + 4 )= *(buff +
buffptr + 0 );
                        *(convbuff + convbuffptr + 5 )= *(buff +
buffptr + 1 );
                        *(convbuff + convbuffptr + 6 )= *(buff +
buffptr + 2 );
                        *(convbuff + convbuffptr + 7 )= *(buff +
buffptr + 3 );
                        *(convbuff + convbuffptr + 8 )= *(buff +
buffptr + 0 );
                        *(convbuff + convbuffptr + 9 )= *(buff +
buffptr + 1 );
                        *(convbuff + convbuffptr + 10 )= *(buff +
buffptr + 2 );
                        *(convbuff + convbuffptr + 11 )= *(buff +
buffptr + 3 );
                        *(convbuff + convbuffptr + 12 )= *(buff +
buffptr + 0 );
                        *(convbuff + convbuffptr + 13 )= *(buff +
buffptr + 1 );
                        *(convbuff + convbuffptr + 14 )= *(buff +
buffptr + 2 );
                        *(convbuff + convbuffptr + 15 )= *(buff +
buffptr + 3 );

                        buffptr += 4;
                        convbuffptr += 16;
                }

                return convbuffptr;
        }

        if ( ( buffsize > 640-20 ) && ( buffsize < 640+20 ) ) // 16k
        {
                while( buffptr+1 < buffsize )
                {
                        *(convbuff + convbuffptr + 0 )= *(buff +
buffptr + 0 );
                        *(convbuff + convbuffptr + 1 )= *(buff +
buffptr + 1 );
                        *(convbuff + convbuffptr + 2 )= *(buff +
buffptr + 2 );
                        *(convbuff + convbuffptr + 3 )= *(buff +
buffptr + 3 );
                        *(convbuff + convbuffptr + 4 )= *(buff +
buffptr + 0 );
                        *(convbuff + convbuffptr + 5 )= *(buff +
buffptr + 1 );
```

```
                        *(convbuff + convbuffptr + 6 )= *(buff +
buffptr + 2 );
                        *(convbuff + convbuffptr + 7 )= *(buff +
buffptr + 3 );


                        buffptr += 4;
                        convbuffptr += 8;

                        if ( (buffptr % 3) == 0 || (buffptr % 25 ) ==
0 )
                        {
                                *(convbuff + convbuffptr + 0 )= *(buff
+ buffptr + 0 );
                                *(convbuff + convbuffptr + 1 )= *(buff
+ buffptr + 1 );
                                *(convbuff + convbuffptr + 2 )= *(buff
+ buffptr + 2 );
                                *(convbuff + convbuffptr + 3 )= *(buff
+ buffptr + 3 );
                                *(convbuff + convbuffptr + 4 )= *(buff
+ buffptr + 0 );
                                *(convbuff + convbuffptr + 5 )= *(buff
+ buffptr + 1 );
                                *(convbuff + convbuffptr + 6 )= *(buff
+ buffptr + 2 );
                                *(convbuff + convbuffptr + 7 )= *(buff
+ buffptr + 3 );


                                //buffptr += 4;
                                convbuffptr += 8;
                        }
                }

                return convbuffptr;
        }

        if ( ( buffsize > 880-20 ) && ( buffsize < 880+20 ) ) // 22k
        {
                while( buffptr+1 < buffsize )
                {
                        *(convbuff + convbuffptr + 0 )= *(buff +
buffptr + 0 );
                        *(convbuff + convbuffptr + 1 )= *(buff +
buffptr + 1 );
                        *(convbuff + convbuffptr + 2 )= *(buff +
buffptr + 2 );
                        *(convbuff + convbuffptr + 3 )= *(buff +
buffptr + 3 );
```

```
                                        *(convbuff + convbuffptr + 4 )= *(buff +
buffptr + 0 );
                                        *(convbuff + convbuffptr + 5 )= *(buff +
buffptr + 1 );
                                        *(convbuff + convbuffptr + 6 )= *(buff +
buffptr + 2 );
                                        *(convbuff + convbuffptr + 7 )= *(buff +
buffptr + 3 );

                                        buffptr += 4;
                                        convbuffptr += 8;
                                }

                                return convbuffptr;
                        }

                        if ( ( buffsize > 1280-20 ) && ( buffsize < 1280+20 ) ) // 32k
                        {
                                while( buffptr+1 < buffsize )
                                {
                                        *(convbuff + convbuffptr + 0 )= *(buff +
buffptr + 0 );
                                        *(convbuff + convbuffptr + 1 )= *(buff +
buffptr + 1 );
                                        *(convbuff + convbuffptr + 2 )= *(buff +
buffptr + 2 );
                                        *(convbuff + convbuffptr + 3 )= *(buff +
buffptr + 3 );

                                        buffptr += 4;
                                        convbuffptr += 4;

                                        if ( (buffptr % 3) == 0 || (buffptr % 25 ) ==
0 )
                                        {
                                                *(convbuff + convbuffptr + 0 )= *(buff
+ buffptr + 0 );
                                                *(convbuff + convbuffptr + 1 )= *(buff
+ buffptr + 1 );
                                                *(convbuff + convbuffptr + 2 )= *(buff
+ buffptr + 2 );
                                                *(convbuff + convbuffptr + 3 )= *(buff
+ buffptr + 3 );

                                                //buffptr += 4;
                                                convbuffptr += 4;
                                        }
                                }
```

```
                return convbuffptr;
        }

        if ( ( buffsize > 1760-20 ) && ( buffsize < 1760+20 ) ) // 44k
        {
                while( buffptr+1 < buffsize )
                {
                        *(convbuff + convbuffptr + 0 )= *(buff +
buffptr + 0 );
                        *(convbuff + convbuffptr + 1 )= *(buff +
buffptr + 1 );
                        *(convbuff + convbuffptr + 2 )= *(buff +
buffptr + 2 );
                        *(convbuff + convbuffptr + 3 )= *(buff +
buffptr + 3 );

                        buffptr += 4;
                        convbuffptr += 4;
                }

                return convbuffptr;
        }

        while( buffptr+1 < buffsize ) // 88k
        {
                *(convbuff + convbuffptr + 0 )= *(buff + buffptr + 0
);
                *(convbuff + convbuffptr + 1 )= *(buff + buffptr + 1
);
                *(convbuff + convbuffptr + 2 )= *(buff + buffptr + 2
);
                *(convbuff + convbuffptr + 3 )= *(buff + buffptr + 3
);

                buffptr += 8;
                convbuffptr += 4;
        }

        return convbuffptr;
}


ULONG ConvertRate( char* buff, char* convbuff, ULONG buffsize )
{
        ULONG   buffptr = 0;
        ULONG   convbuffptr = 0;
        USHORT  iPulse;
```

10915544.072601

```
        ULONG   tblCount = 0;
        ULONG   ConvRate;

/*      ULONG   tblbuffsize[] = {  224,  444,  884, 1764, 3520, 0 };
        ULONG   tblConvRate[] = { 1792, 1776, 1768, 1764, 1760, 0 };

        ConvRate = 1760;
        while( tblbuffsize[ tblCount ] != 0 )
        {
                if ( buffsize == tblbuffsize[ tblCount ] )
                {
                        ConvRate = tblConvRate[ tblCount ];
                        break;
                }
                tblCount++;
        } */

        ConvRate = 1280;

        //char outstring[1024];
        //sprintf( outstring, "buffsize : %d\n", buffsize);
        //OutputDebugString(outstring);

        while( convbuffptr < ConvRate )
        {
                buffptr = (ULONG)( (double)convbuffptr * buffsize /
ConvRate / 4 ) * 4;

                if ( buffptr < buffsize )
                {
                        iPulse = *(USHORT*)(buff + buffptr + 0);
                }
                else
                {
                        iPulse = *(USHORT*)(buff + buffsize - 4);
                }

                *(convbuff + convbuffptr + 0) = *(((char*)&iPulse)+0);
                *(convbuff + convbuffptr + 1) = *(((char*)&iPulse)+1);

                if ( buffptr < buffsize)
                {
                        iPulse = *(USHORT*)(buff + buffptr + 2);
                }
                else
                {
                        iPulse = *(USHORT*)(buff + buffsize - 2);
                }
                *(convbuff + convbuffptr + 2) = *(((char*)&iPulse)+0);
                *(convbuff + convbuffptr + 3) = *(((char*)&iPulse)+1);

                convbuffptr += 4;
        }
        return convbuffptr;
}


ULONG PCMconvert2( char* buff, char* convbuff, ULONG buffsize )
{
        ULONG   nSrcSamplesPerSec = 32000;
        if ( buffsize >= 320 - 8 && buffsize <= 320 + 8 )
nSrcSamplesPerSec = 8000; // 8k
        if ( buffsize >= 440 - 8 && buffsize <= 440 + 8 )
nSrcSamplesPerSec = 11025; // 11k
        if ( buffsize >= 640 - 8 && buffsize <= 640 + 8 )
nSrcSamplesPerSec = 16000; // 16k
        if ( buffsize >= 880 - 8 && buffsize <= 880 + 8 )
nSrcSamplesPerSec = 22050; // 22k
        if ( buffsize >= 1280 - 8 && buffsize <= 1280 + 8 )
nSrcSamplesPerSec = 32000; // 32k
        if ( buffsize >= 1760 - 8 && buffsize <= 1760 + 8 )
nSrcSamplesPerSec = 44100; // 44k
        if ( buffsize >= 2560 - 8 && buffsize <= 2560 + 8 )
nSrcSamplesPerSec = 64000; // 64k
        if ( buffsize >= 3520 - 8 && buffsize <= 3520 + 8 )
nSrcSamplesPerSec = 88200; // 88k

        WAVEFORMATEX wfSrc;
        memset(&wfSrc, 0, sizeof(wfSrc));
        wfSrc.cbSize = 0;
        wfSrc.wFormatTag = WAVE_FORMAT_PCM; // pcm
        wfSrc.nChannels = 2;
        //wfSrc.nSamplesPerSec = 32000;
        wfSrc.nSamplesPerSec = nSrcSamplesPerSec;
        wfSrc.wBitsPerSample = 16;
        wfSrc.nBlockAlign = wfSrc.nChannels * wfSrc.wBitsPerSample /
8;
        wfSrc.nAvgBytesPerSec = wfSrc.nSamplesPerSec *
wfSrc.nBlockAlign;

        //DWORD dwSrcSamples = wfSrc.nSamplesPerSec;

        WAVEFORMATEX wfPCM;
        memset(&wfPCM, 0, sizeof(wfPCM));
        wfPCM.cbSize = 0;
        wfPCM.wFormatTag = WAVE_FORMAT_PCM; // pcm
        wfPCM.nChannels = 2;
        wfPCM.nSamplesPerSec = 32000;
        wfPCM.wBitsPerSample = 16;
        wfPCM.nBlockAlign = wfPCM.nChannels * wfPCM.wBitsPerSample /
8;
        wfPCM.nAvgBytesPerSec = wfPCM.nSamplesPerSec *
wfPCM.nBlockAlign;


        HACMSTREAM hstr = NULL;
        MMRESULT mmr;
        mmr = acmStreamOpen(&hstr,
                                                        NULL, // any driver
                                                        &wfSrc, // source
format
                                                        &wfPCM, // destination
format
                                                        NULL, // no filter
                                                        NULL, // no callback
                                                        0, // instance data
(not used)

ACM_STREAMOPENF_NONREALTIME); // flags
        if (mmr)
        {
                ::AfxMessageBox( "Error acmStreamOpen" );
        }

        //DWORD dwSrcBytes = dwSrcSamples * wfSrc.wBitsPerSample / 8;
/*      DWORD dwSrcBytes = buffsize;
        DWORD dwDst1Samples = dwSrcSamples * wfPCM.nSamplesPerSec /
wfSrc.nSamplesPerSec;
        DWORD dwDst1Bytes = dwDst1Samples * wfPCM.wBitsPerSample / 8;

        BYTE* pDst1Data = new BYTE [dwDst1Bytes];
        memset(pDst1Data, 0, dwDst1Bytes); */

        ACMSTREAMHEADER strhdr;
        memset(&strhdr, 0, sizeof(strhdr));
        strhdr.cbStruct = sizeof(strhdr);
        strhdr.pbSrc = (unsigned char*)buff;
        //strhdr.cbSrcLength = dwSrcBytes;
        strhdr.cbSrcLength = buffsize;
        strhdr.pbDst = (unsigned char*)convbuff;
        //strhdr.cbDstLength = dwDst1Bytes;
        strhdr.cbDstLength = 200000;
        mmr = acmStreamPrepareHeader(hstr, &strhdr, 0);

        mmr = acmStreamConvert(hstr, &strhdr, 0);
        if (mmr)
        {
                ::AfxMessageBox( "Error acmStreamConvert" );
        }

        acmStreamClose(hstr, 0);

        //::AfxMessageBox( "Converted" );

        return strhdr.cbDstLengthUsed;
}

ULONG   PCMNoConvert( char* buff, char* convbuff, ULONG buffsize )
{
        ULONG convbuffptr = 0;
        memcpy( convbuff, buff, buffsize );
        convbuffptr = buffsize;

        return convbuffptr;
}
```

Appendix B

ManagerDeviceDll.cpp

```
// ManagerDeviceDll.cpp: implementation of the CManagerDeviceDll
class.
//
//////////////////////////////////////////////////////////////////////
///

#include "stdafx.h"
#include "iDroid.h"
#include "ManagerDeviceDll.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
///
// Construction/Destruction
//////////////////////////////////////////////////////////////////////
///

CManagerDeviceDll::CManagerDeviceDll()
{

}

CManagerDeviceDll::~CManagerDeviceDll()
{
        this->FreeDeviceDll();
}

void CManagerDeviceDll::SetDllFileName( CString *pstrDllFileName )
{
        this->m_strDllFile = pstrDllFileName->GetBuffer( 0 );
}

void CManagerDeviceDll::SetParentWindow( HWND hWndParent )
{
        this->m_hWndParent = hWndParent;
}

void CManagerDeviceDll::SetStopPointer( BOOL *pbStop )
{
        this->m_pbStop = pbStop;
}

BOOL CManagerDeviceDll::MakeListUploadID( CList<int, int&>
*pListUploadID )
{
        this->m_ListUploadID.RemoveAll();
        int                     nUploadID;
        POSITION        posListUploadID =
pListUploadID->GetHeadPosition();
        while ( posListUploadID )
        {
                nUploadID = pListUploadID->GetNext( posListUploadID
);
                this->m_ListUploadID.AddTail( nUploadID );
        }

        return TRUE;
}

BOOL CManagerDeviceDll::MakeListUploadFiles( CList<CString,
CString&> *pListUploadFiles )
{
        this->m_ListUploadFiles.RemoveAll();

        CString         strUploadFile;
        POSITION        posListUploadFiles =
pListUploadFiles->GetHeadPosition();
        while ( posListUploadFiles )
        {
                strUploadFile = pListUploadFiles->GetNext(
posListUploadFiles );
                this->m_ListUploadFiles.AddTail( strUploadFile );
        }

        return TRUE;
}

void CManagerDeviceDll::SetDeleteAllOption( BOOL bDeleteAll )
{
        this->m_bDeleteAllBeforeUpload = bDeleteAll;
}

BOOL CManagerDeviceDll::LoadDeviceDll( void )
{
        if ( this->m_hDeviceDll != NULL )
        {
                this->FreeDeviceDll();
        }

        this->m_hDeviceDll = ::LoadLibrary(
this->m_strDllFile.GetBuffer( 0 ) );
        if ( this->m_hDeviceDll == NULL )
        {
                // Fail to Load Dll
                return FALSE;
        }

        return TRUE;
```

```
}

BOOL CManagerDeviceDll::FreeDeviceDll( void )
{
        if ( this->m_hDeviceDll == NULL )
        {
                return TRUE;
        }

        // Free Dll
        ::FreeLibrary( this->m_hDeviceDll );
        this->m_hDeviceDll = NULL;

        return TRUE;
}

BOOL CManagerDeviceDll::UploadToDevice( void )
{
        DM_UPLOAD_TO_DEVICE     *pDMUploadToDevice;
        pDMUploadToDevice = (DM_UPLOAD_TO_DEVICE
*)::GetProcAddress( this->m_hDeviceDll, "UploadToDevice" );

        if ( pDMUploadToDevice == NULL )
        {
                return FALSE;
        }

        return (*pDMUploadToDevice)( this->m_hWndParent,
this->m_pbStop, &this->m_ListUploadID, &this->m_ListUploadFiles,
this->m_bDeleteAllBeforeUpload );
}
```

Appendix C

iDroid.cpp

```
// iDroid.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "iDroid.h"
#include "iDroidDlg.h"

#include "GlobalVariables.h"
#include "GetOSVersion.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

//////////////////////////////////////////////////////////////////////
//////////
// CIDroidApp

BEGIN_MESSAGE_MAP(CIDroidApp, CWinApp)
        //{{AFX_MSG_MAP(CIDroidApp)
        //}}AFX_MSG_MAP
        ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()

//////////////////////////////////////////////////////////////////////
//////////
// CIDroidApp construction

CIDroidApp::CIDroidApp()
{
        // TODO: add construction code here,
        // Place all significant initialization in InitInstance
}

//////////////////////////////////////////////////////////////////////
//////////
// The one and only CIDroidApp object

CIDroidApp theApp;

//////////////////////////////////////////////////////////////////////
//////////
// CIDroidApp initialization

BOOL CIDroidApp::InitInstance()
{
        // by tuttii for Single Instance
        this->m_hMutexForSingleInstance = ::CreateMutex( NULL,
TRUE, _T( "Mutex for Single Instance - iDroid" ) );
        if ( ::GetLastError() == ERROR_ALREADY_EXISTS )
        {
                HWND    hWnd    = ::FindWindow( "iDroid Client", 
NULL );
                if ( hWnd )
                {
                        ::ShowWindow( hWnd, SW_SHOW );
                        ::BringWindowToTop( hWnd );
                        ::SetForegroundWindow( hWnd );
                }
                return FALSE;
        }

        // by tuttii
        // _T( "iDroid" ) : company name
        // String Table( in Resource ) AFX_IDS_APP_TITLE : 
application name
        this->SetRegistryKey( _T( "iDroid" ) );

        // by tuttii for ole drag and drop
        if ( !::AfxOleInit() )
        {
                ::AfxMessageBox( CString( 
(LPCSTR)IDS_ERROR_IDROID_FAIL_AFXOLEINIT ).GetBuffer( 0 ) );
                return FALSE;
        }

        if (!AfxSocketInit())
        {
                AfxMessageBox(IDP_SOCKETS_INIT_FAILED);
                return FALSE;
        }

        this->m_nClipboardFormatSharedLink      = 
::RegisterClipboardFormat( _T( "iDroidSharedLink" ) );

        AfxEnableControlContainer();

        // Standard initialization
        // If you are not using these features and wish to reduce 
the size
        //  of your final executable, you should remove from the 
following
        //  the specific initialization routines you do not need.

#ifdef _AFXDLL
        Enable3dControls();                     // Call this when 
using MFC in a shared DLL
#else
        Enable3dControlsStatic();       // Call this when linking 
to MFC statically
#endif
```

10921659 .072601

```
	// by tuttii : Get OS Version
	this->m_nOSVersion	= ::GetOSVersion();
	if ( this->m_nOSVersion < 0 )
	{
		::AfxMessageBox( CString(
(LPCSTR)IDS_ERROR_IDROID_FAIL_GETOSVERSION ).GetBuffer( 0 ) );
		return FALSE;
	}

	// by tuttii
	TCHAR	szCurrentDirectory[ MAX_PATH ];
	if ( !::GetCurrentDirectory( MAX_PATH, szCurrentDirectory )
)
	{
		return FALSE;
	}
	::SetIDroidPath( szCurrentDirectory );

	CIDroidDlg dlg;
	m_pMainWnd = &dlg;
	int nResponse = dlg.DoModal();
	if (nResponse == IDOK)
	{
		// TODO: Place code here to handle when the dialog
is
		//  dismissed with OK
	}
	else if (nResponse == IDCANCEL)
	{
		// TODO: Place code here to handle when the dialog
is
		//  dismissed with Cancel
	}

	// by tuttii for Single Instance
	::CloseHandle( this->m_hMutexForSingleInstance );

	// Since the dialog has been closed, return FALSE so that
we exit the
	//  application, rather than start the application's
message pump.
	return FALSE;
}

BOOL CIDroidApp::InitApplication()
{
	// TODO: Add your specialized code here and/or call the
base class
	WNDCLASS wc;

	wc.style			= CS_DBLCLKS | CS_SAVEBITS
| CS_BYTEALIGNWINDOW;
```

```
        wc.lpfnWndProc              = DefDlgProc;
        wc.cbClsExtra               = 0;
        wc.cbWndExtra               = DLGWINDOWEXTRA;
        wc.hInstance                = ::AfxGetInstanceHandle();
        wc.hIcon                            = this->LoadIcon(
IDR_MAINFRAME );
        wc.hCursor                          = ::LoadCursor( NULL,
IDC_ARROW );
        wc.hbrBackground            = (HBRUSH)COLOR_WINDOW + 1;
        wc.lpszMenuName             = NULL;
        wc.lpszClassName            = "iDroid Client";
        ::RegisterClass( &wc );

        return CWinApp::InitApplication();
}

UINT CIDroidApp::GetSharedLinkFormat( void )
{
        return this->m_nClipboardFormatSharedLink;
}
```

Appendix D

FetchListCtrl.cpp

```
// FetchListCtrl.cpp : implementation file
//

#include "stdafx.h"
#include "iDroid.h"
#include "FetchListCtrl.h"
#include "GlobalMessage.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////
//////////
// CFetchListCtrl

CFetchListCtrl::CFetchListCtrl()
{
        this->m_bDragging               = FALSE;
        this->m_nOldClientWidth = -1;
}

CFetchListCtrl::~CFetchListCtrl()
{
        this->m_font.DeleteObject();
}

BEGIN_MESSAGE_MAP(CFetchListCtrl, CListCtrl)
        //{{AFX_MSG_MAP(CFetchListCtrl)
        ON_WM_KEYDOWN()
        ON_NOTIFY_REFLECT(NM_DBLCLK, OnDblclk)
        ON_NOTIFY_REFLECT(LVN_ITEMCHANGED, OnItemchanged)
        ON_NOTIFY_REFLECT(LVN_BEGINDRAG, OnBegindrag)
        ON_WM_MOUSEMOVE()
        ON_WM_LBUTTONUP()
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////
//////////
// CFetchListCtrl message handlers

void CFetchListCtrl::InitListCtrl( BOOL bShowGrid )
{
        // Set Font ...
        this->SetNewFont();

        this->DeleteAllItems();
```

```
	// Set Style ...
	DWORD	dwExtendedStyle = LVS_EX_FULLROWSELECT;
	if ( bShowGrid )
	{
		dwExtendedStyle |= LVS_EX_GRIDLINES;
	}
	this->ModifyStyle( LVS_TYPEMASK, LVS_REPORT |
LVS_SHOWSELALWAYS );
	//this->SetExtendedStyle( this->GetExtendedStyle() |
LVS_EX_FULLROWSELECT | LVS_EX_GRIDLINES );
	this->SetExtendedStyle( this->GetExtendedStyle() |
dwExtendedStyle );


	// Set Color ...
	this->SetBkColor( RGB( 255, 255, 255 ) );
	this->SetTextColor( RGB( 0, 0, 0 ) );
	this->SetTextBkColor( RGB( 255, 255, 255 ) );


	// Set Header ...
	CRect	rectListCtrl;
	this->GetWindowRect( rectListCtrl );

	LV_COLUMN	lvcolumn;
	int			i;
	CString		strHeader[ NUM_COLUMN ];
	strHeader[ 0 ].LoadString( IDS_FETCHLISTITEM_CLASS );
	strHeader[ 1 ].LoadString( IDS_FETCHLISTITEM_TITLE );
	strHeader[ 2 ].LoadString( IDS_FETCHLISTITEM_DESCRIPTION );
	strHeader[ 3 ].LoadString( IDS_FETCHLISTITEM_DURATION );
	strHeader[ 4 ].LoadString( IDS_FETCHLISTITEM_STATUS );
	for ( i=0; i<NUM_COLUMN; i++ )
	{
		lvcolumn.mask		= LVCF_FMT | LVCF_SUBITEM |
LVCF_TEXT | LVCF_WIDTH;
		if ( i == 3 )
		{
			lvcolumn.fmt	= LVCFMT_RIGHT;
		}
		else
		{
			lvcolumn.fmt	= LVCFMT_LEFT;
		}
		lvcolumn.pszText	= strHeader[ i ].GetBuffer(
0 );
		lvcolumn.iSubItem	= i;
		lvcolumn.cx			= int( double(
(double)rectListCtrl.Width() * g_fColumnWidthRatio[ i ] ) + double(
0.5 ) );
		this->InsertColumn( i, &lvcolumn );
	}
```

```
}

void CFetchListCtrl::SetNewFont( void )
{
//      return;

        //CFont *currentfont;
        //currentfont = this->GetFont();
        LOGFONT logfont;
        //currentfont->GetLogFont( &logfont );
        ::memset( &logfont, 0, sizeof( LOGFONT ) );

        CClientDC       dcFetchListCtrl( this );

        //logfont.lfHeight                              = 89;
        logfont.lfHeight                                = 90;
        //logfont.lfHeight      = -MulDiv( 8, GetDeviceCaps(
dcFetchListCtrl, LOGPIXELSY ), 72 );
        //logfont.lfCharSet                             = HANGUL_CHARSET;
        logfont.lfCharSet                               = DEFAULT_CHARSET;
        //logfont.lfPitchAndFamily      = 34;
        ::strcpy( logfont.lfFaceName, "Arial" );

        //this->m_font.CreatePointFont( 80, "Arial",
&dcFetchListCtrl );
        this->m_font.DeleteObject();
        this->m_font.CreatePointFontIndirect( &logfont,
&dcFetchListCtrl );
        this->SetFont( &this->m_font );

//      currentfont = this->GetFont();
//      currentfont->GetLogFont( &logfont );
}

void CFetchListCtrl::InsertFetchListCtrlItem( CFetchListCtrlItem
tempFetchListCtrlItem, int nIndex )
{
        BOOL    bEnsureVisible  = FALSE;
        if ( nIndex == -1 )
        {
                nIndex = this->GetItemCount();
                bEnsureVisible  = TRUE;
        }

        LV_ITEM li;
        li.mask                 = LVIF_TEXT;
        li.state                = 0;
        li.stateMask    = 0;

        //li.iItem = this->GetItemCount();
        li.iItem                = nIndex;
        // Class
        li.iSubItem             = 0;
        li.pszText              =
tempFetchListCtrlItem.m_strClass.GetBuffer( 0 );
        li.cchTextMax    = MAX_PATH;
        this->InsertItem( &li );

        // Title
        li.iSubItem             = 1;
        li.pszText              =
tempFetchListCtrlItem.m_strTitle.GetBuffer( 0 );
        this->SetItem( &li );

        // Description
        li.iSubItem             = 2;
        li.pszText              =
tempFetchListCtrlItem.m_strDescription.GetBuffer( 0 );
        this->SetItem( &li );

        // Duration
        li.iSubItem             = 3;
        li.pszText              =
tempFetchListCtrlItem.m_strDuration.GetBuffer( 0 );
        this->SetItem( &li );

        // Status
        li.iSubItem             = 4;
        li.pszText              =
tempFetchListCtrlItem.m_strStatus.GetBuffer( 0 );
        this->SetItem( &li );

        if ( bEnsureVisible )
        {
                this->EnsureVisible( nIndex, FALSE );
        }
}

void CFetchListCtrl::FitToParentWindow( CRect rectParentClient )
{
        BOOL    bVertScroll;

        // New Window Size
        CRect   rectNew;
        rectNew.left     = 6;
        rectNew.right    = rectParentClient.Width() - 6;
        rectNew.top             = 127;
        rectNew.bottom   = rectParentClient.Height() - 25;

        // Get Header's WindowRect
        CRect   rectHeader;
        CWnd    *pHeader        = this->GetHeaderCtrl();
        pHeader->GetWindowRect( &rectHeader );
        // New Client Width, Height
        int                 nNewClientWidth          = rectNew.Width() -
( ::GetSystemMetrics( SM_CXBORDER ) * 2 );
        int                 nNewClientHeight         = rectNew.Height()
- ( ::GetSystemMetrics( SM_CYBORDER ) * 2 );

        int                 nItemCount       = this->GetItemCount();
        if ( nItemCount > 0 )
        {
                // Get One Item Rect
                CRect   rectItem;
                this->GetItemRect( 0, &rectItem, LVIR_BOUNDS );
                //bVertScroll    = nItemCount > ( ( nNewClientHeight
 rectHeader.Height() - ( ::GetSystemMetrics( SM_CYBORDER ) * 2 ) )
 rectItem.Height() );
                bVertScroll      = nItemCount > ( ( nNewClientHeight
 rectHeader.Height() ) / rectItem.Height() );
                if ( bVertScroll )
                {
                        // Subtract (Vertical Scroll Bar's Width)
from (New Client Width)
                        nNewClientWidth -= ::GetSystemMetrics(
SM_CXVSCROLL );
                }
        }

        this->SetRedraw( FALSE );
        if ( this->m_nOldClientWidth > nNewClientWidth )
        {
                this->FitColumnWidth( nNewClientWidth );
                this->MoveWindow( &rectNew, TRUE );
        }
        else if ( this->m_nOldClientWidth < nNewClientWidth )
        {
                this->MoveWindow( &rectNew, TRUE );
                this->FitColumnWidth( nNewClientWidth );
        }
        else
        {
                this->MoveWindow( &rectNew, TRUE );
        }
        this->SetRedraw( TRUE );
        if ( ( bVertScroll ) && ( this->m_nOldClientWidth !=
nNewClientWidth ) )
        {
                this->Invalidate();
        }

        CRect   rectClient;
        this->GetClientRect( &rectClient );
        this->m_nOldClientWidth = rectClient.Width();
```

```
}

void CFetchListCtrl::FitColumnWidth( int nWidthClient )
{
        int     nTotalWidth;
        int     nColumnWidth;
        nTotalWidth     = nWidthClient;

        for ( int i = 0; i < NUM_COLUMN; i++ )
        {
                nColumnWidth    = int( double( (double)nTotalWidth
* g_fColumnWidthRatio[ i ] ) + double( 0.5 ) );
                this->SetColumnWidth( i, nColumnWidth );
        }
}

BOOL CFetchListCtrl::OnNotify(WPARAM wParam, LPARAM lParam,
LRESULT* pResult)
{
        // TODO: Add your specialized code here and/or call the
base class

        static  BOOL    bChangedByTrack = FALSE;
        int                             i, nTotalWidth, nRatio;

        NMHEADER        *pNMHeader                      = (NMHEADER
*)lParam;
        switch ( pNMHeader->hdr.code )
        {
                case HDN_ENDTRACKW :
                case HDN_ENDTRACKA :
                        bChangedByTrack                 = TRUE;
                        break;
                case HDN_ITEMCHANGEDW :
                case HDN_ITEMCHANGEDA :
                        if ( bChangedByTrack )
                        {
                                bChangedByTrack         = FALSE;
                                nTotalWidth                     =
0;
                                for ( i = 0; i < NUM_COLUMN; i++ )
                                {
                                        nTotalWidth     +=
this->GetColumnWidth( i );
                                }
                                for ( i = 0; i < NUM_COLUMN; i++ )
                                {
                                        nRatio  = int( ( ( double(
this->GetColumnWidth( i ) ) / double( nTotalWidth ) )

                                        + double( 0.0005 ) )
```

```
                                                    * 1000 );
                                                    g_fColumnWidthRatio[ i ]
= double( nRatio ) / double( 1000 );
                                          }
                                }
                                break;
                      default :
                                break;
          }

          return CListCtrl::OnNotify(wParam, lParam, pResult);
}

void CFetchListCtrl::OnKeyDown(UINT nChar, UINT nRepCnt, UINT
nFlags)

          // TODO: Add your message handler code here and/or call
default

          if ( nChar == VK_DELETE )
          {
                    this->GetParent()->PostMessage(
WM_USER_DELETEKEYLISTCTRL );
          }

          CListCtrl::OnKeyDown(nChar, nRepCnt, nFlags);

void CFetchListCtrl::DisplayDownloadStatus( int nIndex, ULONG
ulProgress, ULONG ulProgressMax )
{
          CString           strProgress;
          strProgress.Format(
IDS_FORMAT_FETCHLISTCTRL_DISPLAYSTATUS_DOWNLOADING, ulProgress,
ulProgressMax );
          this->SetItemText( nIndex, 4, strProgress.GetBuffer( 0 ) );
}

void CFetchListCtrl::DisplayConvertStatus( int nIndex, int nPercent
)
{
          CString           strProgress;
          strProgress.Format(
IDS_FORMAT_FETCHLISTCTRL_DISPLAYSTATUS_CONVERTING, nPercent );
          this->SetItemText( nIndex, 4, strProgress.GetBuffer( 0 ) );
}

void CFetchListCtrl::DisplayUploadStatus( int nIndex, int nPercent
)
{
          CString           strProgress;
          strProgress.Format(
IDS_FORMAT_FETCHLISTCTRL_DISPLAYSTATUS_UPLOADING, nPercent );
        this->SetItemText( nIndex, 4, strProgress.GetBuffer( 0 ) );
}

void CFetchListCtrl::OnDblclk(NMHDR* pNMHDR, LRESULT* pResult)
{
        // TODO: Add your control notification handler code here

        this->GetParent()->PostMessage( WM_USER_DOUBLECLICKLISTCTRL
);

        *pResult = 0;
}

int CFetchListCtrl::GetCurrentSelectedIndex( void )
{
        POSITION        posSelectedItem =
this->GetFirstSelectedItemPosition();
        if ( posSelectedItem == NULL )
        {
                return -1;
        }

        return this->GetNextSelectedItem( posSelectedItem );
}

void CFetchListCtrl::SetAtFetchListCtrlItem( int nIndex,
CFetchListCtrlItem tempFetchListCtrlItem )
{
        LV_ITEM li;
        li.mask                 = LVIF_TEXT;
        li.state                = 0;
        li.stateMask    = 0;

        li.iItem                = nIndex;

        // Class
        li.iSubItem             = 0;
        li.pszText              =
tempFetchListCtrlItem.m_strClass.GetBuffer( 0 );
        li.cchTextMax   = MAX_PATH;
        this->SetItem( &li );

        // Title
        li.iSubItem             = 1;
        li.pszText              =
tempFetchListCtrlItem.m_strTitle.GetBuffer( 0 );
        this->SetItem( &li );

        // Description
        li.iSubItem             = 2;
        li.pszText              =
tempFetchListCtrlItem.m_strDescription.GetBuffer( 0 );
        this->SetItem( &li );

        // Duration
        li.iSubItem             = 3;
        li.pszText              =
tempFetchListCtrlItem.m_strDuration.GetBuffer( 0 );
        this->SetItem( &li );

        // Status
        li.iSubItem             = 4;
        li.pszText              =
tempFetchListCtrlItem.m_strStatus.GetBuffer( 0 );
        this->SetItem( &li );

void CFetchListCtrl::OnItemchanged(NMHDR* pNMHDR, LRESULT* pResult)


        NM_LISTVIEW* pNMListView = (NM_LISTVIEW*)pNMHDR;
        // TODO: Add your control notification handler code here

        this->GetParent()->PostMessage( WM_USER_SELCHANGELISTCTRL
);

        *pResult = 0;


void CFetchListCtrl::OnBegindrag(NMHDR* pNMHDR, LRESULT* pResult)
{
        NM_LISTVIEW* pNMListView = (NM_LISTVIEW*)pNMHDR;
        // TODO: Add your control notification handler code here

        ::SetCursor( ::AfxGetApp()->LoadCursor( IDC_CURSOR_DRAG )
);
        this->SetCapture();
        this->m_bDragging       = TRUE;

        *pResult = 0;
}

void CFetchListCtrl::OnMouseMove(UINT nFlags, CPoint point)
{
        // TODO: Add your message handler code here and/or call
default

        if ( this->m_bDragging )
        {
                CPoint  ptScreen( point );
                this->ClientToScreen( &ptScreen );
                if ( this->GetSafeHwnd() == this->WindowFromPoint(
ptScreen )->GetSafeHwnd() )
```

```
                {
                        ::SetCursor( ::AfxGetApp()->LoadCursor(
IDC_CURSOR_DRAG ) );
                }
                else
                {
                        ::SetCursor(
::AfxGetApp()->LoadStandardCursor( IDC_NO ) );
                }
        }

        CListCtrl::OnMouseMove(nFlags, point);
}

void CFetchListCtrl::OnLButtonUp(UINT nFlags, CPoint point)

        // TODO: Add your message handler code here and/or call
default

        if ( this->m_bDragging )
        {
                ::ReleaseCapture();
                ::SetCursor( ::AfxGetApp()->LoadStandardCursor(
IDC_ARROW ) );
                this->m_bDragging       = FALSE;

                CPoint  ptScreen( point );
                this->ClientToScreen( &ptScreen );
                if ( this->GetSafeHwnd() == this->WindowFromPoint(
ptScreen )->GetSafeHwnd() )
                {
                        this->m_ptDrop  = point;
                        this->OnDropItemToReorder();
                }
        }

        CListCtrl::OnLButtonUp(nFlags, point);
}

void CFetchListCtrl::OnDropItemToReorder( void )
{
        if ( this->GetSelectedCount() < 1 )
        {
                return;
        }

        // Get Insert Point & Index
        CRect   rectItem;
        int             nHeightItem;
        int             nDropIndex;
        this->GetItemRect( 0, &rectItem, LVIR_BOUNDS );
        nHeightItem     = rectItem.Height();
        this->m_ptDrop.x        = 0;
        this->m_ptDrop.y        += ( nHeightItem / 2 );
        nDropIndex      = this->HitTest( this->m_ptDrop );
        if ( nDropIndex == -1 )
        {
                nDropIndex      = this->GetItemCount();
        }

        // make int array of selected index
        int                     *pnSelectedIndex;
        pnSelectedIndex = new int[ this->GetSelectedCount() + 1 ];
        POSITION        posSelected =
this->GetFirstSelectedItemPosition();
        while ( posSelected )
        {
                *pnSelectedIndex        =
this->GetNextSelectedItem( posSelected );
                pnSelectedIndex ++;
        }
        *pnSelectedIndex        = -1;
        pnSelectedIndex         -= this->GetSelectedCount();

        // delete selected items
        int             nIndexSelected;
        posSelected = this->GetFirstSelectedItemPosition();
        while ( posSelected )
        {
                nIndexSelected  = this->GetNextSelectedItem(
posSelected );
                this->DeleteItem( nIndexSelected );
                posSelected = this->GetFirstSelectedItemPosition();
        }

        // Reorder ListCtrl & ListFetchListItem
        this->GetParent()->SendMessage( WM_USER_DROPITEMTOREORDER,
(WPARAM)nDropIndex, (LPARAM)pnSelectedIndex );

        delete[]        pnSelectedIndex;
}

void CFetchListCtrl::SetSelectAll( void )
{
        for ( int i = 0; i < this->GetItemCount(); i++ )
        {
                this->SetItemState( i, LVIS_SELECTED, LVIS_SELECTED
);
        }
}

void CFetchListCtrl::SetShowGrid( BOOL bShow )
{
        if ( bShow )
        {
                this->SetExtendedStyle( this->GetExtendedStyle() |
LVS_EX_GRIDLINES );
        }
        else
        {
                this->SetExtendedStyle( this->GetExtendedStyle() &
~LVS_EX_GRIDLINES );
        }
}
```

Appendix E

```
// DlgInstantRecording.cpp : implementation file
//

#include "stdafx.h"
#include "iDroid.h"
#include "DlgInstantRecording.h"
#include "FileUtil.h"
#include "iDroidDlg.h"
#include "CaptureDriver.h"
#include "tuttiiLog.h"
#include "ThreadConvert.h"
#include "GlobalMessage.h"
#include "GetOSVersion.h"
#include "registryutil.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

#define ID_TIMER_INSTANTRECORDING_ELAPSE        100
#define ELAPSE_INSTANTRECORDING                         1000

#define BUTTONSTATE_STARTRECORDING                      1
#define BUTTONSTATE_STOPRECORDING                       2
#define BUTTONSTATE_CANCELCONVERTING            3
////////////////////////////////////////////////////////////////////
//////////
// CDlgInstantRecording dialog


CDlgInstantRecording::CDlgInstantRecording(CWnd* pParent /*=NULL*/)
        : CDialog(CDlgInstantRecording::IDD, pParent)
{
        //{{AFX_DATA_INIT(CDlgInstantRecording)
                // NOTE: the ClassWizard will add member
initialization here
        //}}AFX_DATA_INIT

        this->m_nButtonState                            =
BUTTONSTATE_STARTRECORDING;
        this->m_bAddToFetchListAfterConvert     = FALSE;
}


void CDlgInstantRecording::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CDlgInstantRecording)
                // NOTE: the ClassWizard will add DDX and DDV calls
here
	//}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CDlgInstantRecording, CDialog)
	//{{AFX_MSG_MAP(CDlgInstantRecording)
	ON_BN_CLICKED(IDC_BUTTON_STARTRECORDING,
OnButtonStartrecording)
	ON_WM_TIMER()
	//}}AFX_MSG_MAP
	ON_MESSAGE( WM_USER_CONVERTSUCCESS, OnConvertSuccess )
	ON_MESSAGE( WM_USER_CONVERTFAILED, OnConvertFailed )
	ON_MESSAGE( WM_USER_CONVERTSTATUS, OnConvertStatus )
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////
//////////
// CDlgInstantRecording message handlers

BOOL CDlgInstantRecording::OnInitDialog()
{
	CDialog::OnInitDialog();

	// TODO: Add extra initialization here

	// Button Text
	this->SwitchStartStopButton();

	// Set Progress Bar Control's Range
	CProgressCtrl	*pProgressCtrl	= (CProgressCtrl
*)this->GetDlgItem( IDC_PROGRESS_CONVERTING );
	if ( pProgressCtrl == NULL )
	{
		return FALSE;
	}
	pProgressCtrl->SetRange( 0, 100 );

	return TRUE;  // return TRUE unless you set the focus to a
control
	              // EXCEPTION: OCX Property Pages should
return FALSE
}

void CDlgInstantRecording::OnTimer(UINT nIDEvent)
{
	// TODO: Add your message handler code here and/or call
default

	switch ( nIDEvent )
	{
		case ID_TIMER_INSTANTRECORDING_ELAPSE :
```

```
                        this->DisplayElapseTime();
                        break;
        }

        CDialog::OnTimer(nIDEvent);
}

void CDlgInstantRecording::DisplayElapseTime( void )
{
        CTime           timePresent     = CTime::GetCurrentTime();
        CTimeSpan       timeElapse      = timePresent -
this->m_timeStartRecording;
        CString         strElapse       = timeElapse.Format(
IDS_FORMAT_INSTANTRECORDING_ELAPSETIME );
        this->SetDlgItemText( IDC_STATIC_ELAPSETIME,
strElapse.GetBuffer( 0 ) );


void CDlgInstantRecording::SetInstantRecordingElapseTimer( BOOL
bSet )

        static  BOOL    bAlreadySet     = FALSE;

        if ( bSet && !bAlreadySet )
        {
                this->m_timeStartRecording      =
CTime::GetCurrentTime();
                this->SetTimer( ID_TIMER_INSTANTRECORDING_ELAPSE,
ELAPSE_INSTANTRECORDING, NULL );
                bAlreadySet     = TRUE;
        }
        if ( !bSet && bAlreadySet )
        {
                this->KillTimer( ID_TIMER_INSTANTRECORDING_ELAPSE
);
                bAlreadySet     = FALSE;
        }
}

// Button...
void CDlgInstantRecording::OnButtonStartrecording()
{
        switch ( this->m_nButtonState )
        {
                case BUTTONSTATE_STARTRECORDING :
                        this->StartInstantRecording();
                        break;
                case BUTTONSTATE_STOPRECORDING :
                        this->StopInstantRecording();
                        break;
                case BUTTONSTATE_CANCELCONVERTING :
                        this->StopConvert();
                        break;
        }
}

void CDlgInstantRecording::SwitchStartStopButton( void )
{
        CWnd    *pButton        = this->GetDlgItem(
IDC_BUTTON_STARTRECORDING );
        CString strButtonText;
        switch ( this->m_nButtonState )
        {
                case BUTTONSTATE_STARTRECORDING :
                        strButtonText.LoadString(
IDS_INSTANTRECORDING_STARTRECORDING );
                        break;
                case BUTTONSTATE_STOPRECORDING :
                        strButtonText.LoadString(
IDS_INSTANTRECORDING_STOPRECORDING );
                        break;
                case BUTTONSTATE_CANCELCONVERTING :
                        strButtonText.LoadString(
IDS_INSTANTRECORDING_CANCELCONVERTING );
                        break;
        }
        pButton->SetWindowText( strButtonText.GetBuffer( 0 ) );
}

BOOL CDlgInstantRecording::StartInstantRecording( void )
{
        // Check Capture Item, Sound Driver
        if ( ( (CIDroidApp *)::AfxGetApp() )->m_nOSVersion <
WINDOWS_2000 )
        {
                if ( !::CheckSoundDriver() )
                {
                        return FALSE;
                }
        }

        // Change Button State
        this->m_nButtonState                            =
BUTTONSTATE_STOPRECORDING;
        this->SwitchStartStopButton();

        this->m_bAddToFetchListAfterConvert     = FALSE;

        // Start Timer
        this->SetInstantRecordingElapseTimer( TRUE );

        // Start Capture
        ::StartCaptureDriver();
        return TRUE;
}

CString         l_strDescription;
BOOL            l_bAddToFetchList       = FALSE;
UINT CALLBACK OfnHookProcedure( HWND hDlg, UINT uiMsg, WPARAM
wParam, LPARAM lParam )
{
        OFNOTIFY        *pOfNotify = (LPOFNOTIFY)lParam;
        WORD            wLow, wHigh;
        switch ( uiMsg )
        {
                case WM_NOTIFY :
                        // On "OK" Button
                        if ( pOfNotify->hdr.code == CDN_FILEOK )
                        {
                                // Get Description
                                ::GetDlgItemText( hDlg,
IDC_EDIT_DESCRIPTION, l_strDescription.GetBuffer( MAX_PATH ),
MAX_PATH );
                                l_strDescription.ReleaseBuffer();
                        }
                        // On InitDialog
                        else if ( pOfNotify->hdr.code ==
CDN_INITDONE )
                        {
                                // Init variables about
"Description"
                                l_strDescription.Empty();
                                l_bAddToFetchList       = FALSE;
                                // Uncheck CheckBox
                                ::SendMessage( ::GetDlgItem( hDlg,
IDC_CHECK_ADDTOFETCHLIST ), BM_SETCHECK, BST_UNCHECKED, 0 );
                                // Diable Description
                                ::EnableWindow( ::GetDlgItem( hDlg,
IDC_EDIT_DESCRIPTION ), FALSE );
                        }
                        break;
                case WM_COMMAND :
                        wLow    = LOWORD( wParam );
                        wHigh   = HIWORD( wParam );
                        // On Clicked CheckBox
                        if ( ( wLow == IDC_CHECK_ADDTOFETCHLIST )
&& ( wHigh == BN_CLICKED ) )
                        {
                                LRESULT checkstate      =
::SendMessage( (HWND)lParam, BM_GETCHECK, 0, 0 );
                                if ( checkstate == BST_CHECKED )
                                {
                                        l_bAddToFetchList       =
TRUE;
                                        ::EnableWindow(
```

Appendix F

```
//
// Main process of In-Content AD
// Pseudo Code based on C++ gramma
// by iDroid
//



//
// Main function to receive orders
// from the main component of Client to perform
// the merge of Content and Ad
//
main( inputValues )
{
        // initialization for merging process
        do_initialize();

        // analyze received orders and check for errors
        do_checkInputValues();

        // analyze information of original content and save
        storeContentProperty();

        // analyze ad or ads to be merged, and save
        storeAdProperty();

        // function to merge content and ads
        ConcatenateFile( Content, AdList, Position, etcData );

        // send the ad- merged new content to the component that requested it.
        do_reportResult();

        // finalization of merging process
        do_finalize();
}
```

```
//
// main function to merge ads to contents
//
ConcatenateFile( Content, AdList, Position, etcData )
{
        // inspect contents that will merge with ads
        do_checkContent();
        // retrieve saved information of contents
        get_ContentProperty();

        // separate other information that has been attached by the content's
        // original media format and temporarily save it
        remove_header_and_tag_from_Content();
        // analyze the content that actually merges with the ads
        analyze_Content();

        // repeat process until all ads are merged
        while( ! endof_ADList )
        {
                // analyze the location of the content where the ad will merge
                find_Position_in_Content();
                // perform actual merging of the ad to the content
                merge_Ad_into_Content();
        }

        // merge other info such as headers and tags, which are in the original format,
        // to the new ad-merged content
        add_header_and_tag_to_Content();
        // update and save the info of ad- merged content
        update_other_info();

        // return ad-merged contents
        return result_Content;
}
```

09714607.72601

```
//
// perform actual function of merging content and ad
//
merge_Ad_into_Content()
{
        // retrieve location info of where ad will be merged on the content
        getPositionInfo();

        // analyze format of content
        analyze_media_format_of_Content();

        // change and customize the content format so that it will be easier to merge the
        //ad
        convert_Content_to_custom_format();

        // merge ad and content at appropriate location
        merge_Ad_process();

        // change the format of the ad-merged content to that of the original content
        encode_Content_to_original_format();

        // return ad-merged content
        return result;
}


//
// detail function of merging ad to the content
//
merge_Ad_process()
{
        // save : from the beginning of the original content to the beginning
        //of the ad that is merged
        store_original_to_new( start_of_original_content, start_of_content_to_merge );
        // save :the ad and the original content in a mixed format
        store_ad_mixed_to_new( start_of_Ad, end_of_Ad, start_of_content_to_merge,
end_of_content_to_merge );

        // save: from the end of the ad, to the end of the content, visa versa.
        store_original_to_new( end_of_content_to_merge, end_of_original_content );

        // return newly made content
        return newContent;
}

// end of pseudo code
```

09165644.072601

4

United States Patent & Trademark Office

Office of Initial Patent Examination -- Scanning Division

09515544.072601

Application deficiencies found during scanning:

☐ Page(s)__________ of ____________________________ were not present for scanning.
(Document title)

☐ Page(s)__________ of ____________________________ were not present for scanning.
(Document title)

☑ ***Scanned copy is best available.*** Drawings